

status

N E W S L E T T E R

**NO,
IT'S NOT
QUARTERLY!**

PRESIDENT	
BUCK MADDREY.....	464-2100
VICE PRESIDENT	
STAN HARRISON.....	486-3882
TREASURER	
DAVID LEVY.....	479-2333
SECRETARY	
DICK LITCHFIELD.....	468-6964
PROGRAM COORDINATOR	
JIM PARKS.....	497-6526
LIBRARIAN	
CHUCK SARGENT.....	422-5691
MEMBERSHIP CHAIRMAN	
RON JOHNSON.....	623-4362
SYSOP	
DOUG BOYNTON.....	468-6242
EDITOR	
GENE RODRIGUEZ III.....	463-2453

STATUS BBS

1 - 804 - 468 - 1096

ADVERTISING RATES: The following rates refer to a one month presentation of a given ad, other rates may apply to extended presentations. All advertisements must be camera ready, and must be delivered to the Editor no later than one week prior to publication. Make all checks payable to STATUS.

FULL PAGE
(5" X 7") \$25.00

BUSINESS CARD
(3" X 5") \$10.00

Editor's File

Deep Philosophy

Sometimes, things happen for a single reason and sometimes they happen for a multitude of reasons. Hmmm, that seems obvious. Let's try again.

~~~~~  
**Remember, it's not  
just a job, it's  
an adventure...**  
~~~~~

You know how frequently you'll find yourself in a situation where no answer seems to be the right one? Ummm, too vague. Oh, to hell with it!

In case you don't already know, I'm resigning as Newsletter editor. Having received a new position at work which creates a lot more strain (and responsibility) for me, and with the warm weather and associated projects around the house, I just don't have the time to keep up the quality or schedule that the Newsletter deserves. Part of this has manifested itself in the WAY behind schedule issue that you now hold in your hands, and for that I apologize. Hopefully, a new editor will be

able to get it back on a regular basis.

After three and a half years as editor (My God, has it really been that long?) I've learned a lot about publishing (the desktop kind), my computer (both XL and ST) and the great group of people who've made STATUS the outstanding club it has become. If YOU'D like to have a hand in molding the image that other groups have of STATUS, then contact any member of the Executive Committee and let them know.

Remember, it's not just a job, it's an adventure...

Fore!

Just received a flier in the mail the other day from Accolade, the makers of Mean 18 (one of the best games ever created, by the way). The flier was to announce the release of three (that's right, three) new course disks! The "Famous Course" series continues with volumes II, III and IV with courses including: Turnberry (Scotland), Inverness Club, Harbour Town, Doral, Olympic, Las Colinas, Kapalua, Muirfield (Scotland), and Castle Pines!

ST BLITTER

By Mike Fulton

OmJuce: November, 1986

Atari is on the verge of finally releasing the long awaited blitter chip for the ST computers. But with many ST owners, there is confusion about just what exactly this blitter chip does. What is a blitter? Well, I'll try to explain. Pardon me if it gets a little hard to follow at times, but I'll try to make it easy...

A blitter chip, is a chip which blits...

A blitter chip is a chip which blits. At this you may say, "Even I knew that, but what does it mean?" The term Blit is short for another term, Bit Block Transfer (Bit-BLT). This is a fancy name for moving the contents of one part of computer memory to another part, or transferring a block of bits. The way it does this is special. Instead of working on whole bytes of memory, blitting allows the individual bits of each byte to be accessed separately. This way, a bit-logic operation can also be performed on each bit during the transfer. This is very important when working with graphics memory, since

one byte of memory may represent several points on the display screen.

The ST already has blitting routines written in software. These routines are used for many purposes, including most graphics output to screen. Even printing text on the screen is done by blitting the character data from the font to the screen. Also, every time you move a window around on the desktop, you are making the system blit the screen memory to another part of the screen memory.

Because the ST's blitting routines are in software, the 68000 cpu must do everything all by itself. This works fine, but a custom designed blitter chip can do this faster. Once the 68000 sets up the right parameters, the blitter chip can do the bit operation without any more help, doing it faster and freeing the 68000 for other things.

One important thing to know is when the blitter chip comes out it will have new TOS ROM chips with it. In the new ROM chips the current software blitter routines will be replaced with routines which just pass the necessary information to the blitter chip. This way, all programs which

used the current software blit routines in the normal manner will still work with the blitter chip, but at a higher speed.

Programs which do not use the current software blitting routines will not experience any difference in speed with the blitter chip. This is important since most current games with a lot of animation probably use their own routines instead of the built-in ones. These games will not be speeded up beyond the point of playability.

Secrets of ST Bit-Block Transfer

Since the major uses of blitting are with the graphics and text output, the GEM blit routines are designed to be easily used in this fashion. The Atari ST also has a set of low-level routines which are designed specifically to be used by the GEM graphics routines,

including blitting, which is known as the Line-A Interface. Using the blit routines is easier done with GEM, because the Line-A Interface can usually only be accessed via machine language. For all but the most speed intensive uses, there should not be any practical differences in performance.

Regardless of which method you choose, most of the following information applies to both methods of accessing the blit functions.

Before blitting to or from an area of memory, you first define the area as though it was a graphics screen, or raster form. There is a special memory structure used by the ST for just this purpose, called a Memory Form Definition Block, or MFDB. The MFDB is set up like this (a long is 4 bytes of memory, and a word is 2 bytes):

MFDB= long ADDRESS
word X_RES
word Y_RES
word WORD_WID
word FORMAT
word PLANES
word RES_1
word RES_2
word RES_3

Address of form
Width, in pixels
height, in pixels
Width, in words
Memory format flag
of bit planes
Reserved value #1
Reserved value #2
Reserved value #3

The ADDRESS parameter should be set to the memory address of the top left corner of the raster form. When using the GEM VDI blit routines, if this address is set to 0, then the routines use the actual display screen for the form and GEM fills in the rest of the parameters by itself. Don't do this when using the Line-A routines, because it will not work.

The X_RES and Y_RES parameters are used to define the height and width, in pixels, of a memory form. This can vary quite a bit depending on what sort of memory form is involved. For example, the ST's system font data is a memory form 2048 pixels wide, and either 8 to 16 pixels tall (depending on the screen resolution used). That's a very wide, short form!

The WORD_WID parameter is used to hold the width, in words, of form. This is to determine how much memory is used by each row. This can be determined by taking the X_RES parameter and multiplying it by the PLANES parameter, and then dividing the result by the word size (16 bits). You use the result rounded up to the next integer.

Before discussing the FORMAT parameter, let's skip to the PLANES parameter. This is used to specify how many bit planes are used in the memory form. What's a bit plane? Well, it's not easy to explain, but here goes! Listen carefully.

The ST's screen memory is arranged in bit planes. The number of colors available depends on how many bit planes are used. With 4 bit planes, there is one bit in each plane for one pixel on the screen. Since 4 bits can contain 16 possible combinations, this is how many colors the screen can have with 4 bit planes. Thus, monochrome resolution has 1 bit plane with 2 colors (white or black); medium has 2 bit planes and 4 colors; and low resolution has 4 bit planes with 16 colors. If the ST could have 5 bit planes, you could get 32 colors, and so on. The video chip of the ST grabs one bit from each bit plane and adds them together to get the color to display for each pixel on the screen.

The FORMAT parameter is used to tell the bit routines how the bit planes of the form are arranged. Because of the way the ST's video chip works,

the bit planes are mixed together in a certain way. But this is only the way the ST works. Since the GEM blit routines must work on different computers, and each computer might have its own method of arranging screen memory, there are two ways this parameter can be set. The first is called Device Specific. This means the memory for the bit planes is arranged in the way the device (the ST video chip in this case) uses it. On an IBM running GEM, this means the memory is [not] arranged the way the computer's graphics card wants it.

The second format is called Standard Format. This means all the bit planes are separated from one another. All of the first bit plane's memory comes first, followed by all the second bit plane's memory, and so on. One of the GEM blit routines is designed to convert from one format to the other. When blitting is always done within the same system, as for drawing programs or animation, this parameter is almost always used as Device Specific. But when there's a need to move data across different systems, it is changed to the Standard Format first.

On the ST, Device Specific is usually used for everything except for displaying fill patterns, character font data, and drawing icons on the desktop, since these things are sometimes shared across different systems.

The last three parameters, RES_1, RES_2, and RES_3, are all reserved for future expansion, and should always be set to zero.

OK, that's pretty much what you need to know about the parameters for the blit routines, now on to actually using them! After setting up both your source and destination forms, you set up an array which contains the top left and bottom right corners of both the area you are blitting from and the area you are blitting into. Both areas must be the same size, or you will get strange, unpredictable results...

An Update on ST Ramdisks

By Stephen D. Eitelman

Current Notes: March, 1987

Intro

A RAM disk is a portion of the computer's Random Access

Memory (RAM) that has been fenced off to function as a disk drive. Software is used to fool the operating system into thinking that portion of memory is actually a floppy drive. There are two primary advantages to such a scheme. First, a RAM disk is fast. There are no mechanical delays in positioning the head or waiting for the disk to be spun to the proper sector. Secondly, there is no mechanical wear of the drive mechanism or disk (remember, they don't actually exist). This is important in applications that are especially disk intensive such as some word processors, some games and software development that requires many repetitions of the edit-compile-link-run-crash cycle.

~~~~~  
**First, a RAM disk is  
FAST...**  
~~~~~

The RAM disk is even faster than the Supra 20MB hard disk (on which these comments are based), but not substantially so. The real advantage of the RAM disk when a hard disk is in use, is to save mechanical wear on the hard disk. These things are delicate creatures and they

always sound as if they are going to fly apart when the disk is being accessed. The RAM disk makes me feel much more comfortable with its total silence and I'm sure the hard drive will last much longer this way.

RAM Disk Compatibility

Hard disks, such as the Supra 20MB drive, permit the drive to be partitioned (organizationally split) into as many as four separate drives, identified as C, D, E, and F. Even if only one drive (C) is selected, the controller still appears to allow for the remaining three to be selected at a later time. If a RAM disk is installed that utilizes drive identifier D, there will be a conflict between the RAM disk and the hard drive, regardless of the partitioning. The result will be either a warning that the drive does not exist when an attempt is made to access drive D, or a loss of access to the hard disk, drive D.

RAM Disk Requirements

What is clearly needed then, is a RAM disk that is compatible with the hard disk. It is also desirable that the RAM disk be immune to the reset switch and resolution

changes. This way, when a program bombs and locks out the keyboard, or a resolution change is made, the files will still be in the RAM disk after a reset and boot. Additionally, the amount of memory assigned to the RAM disk should be adjustable to allow for different size computer memories and to permit the ratio of working file memory to RAM disk memory to be varied as a function of the application. A RAM disk should also be compatible with an automatic loader, so that files currently being used can be loaded automatically at boot-up.

Which RAM Disk?

Twenty six RAM disk programs were examined, two commercial and the remainder public domain. The sources for the public domain RAM disks were the Current Notes ST library utility disks, GENie and CompuServe. This selection of Ram disks is not exhaustive (every time I go into CompuServe or GENie, there is another one!) but these twenty six RAM disks seem to be representative. Only nine of them were compatible with the hard disk, of these, three (ETERNAL, YARD and RAMSKIM) are also "reset

proof" and immune to changes in resolution between medium and low on a color monitor. ETERNAL and YARD are also adjustable in size; RAMSKIM (after modification by RAMIMFIX for use with the hard disk) is fixed in size at 500K bytes. The chart summarizes the RAM disks that were found to be compatible with the hard disk. Any RAM disk that was not found to be reset-proof was not tested any further.

RAMDISK	SOURCE	HD COMPAT?	RESET PROOF?	SIZE ADJ?	AUTO LOAD?
RAMSKIM.PRG	PD	Yes	Yes	No	Yes
FASTRAM.TTP	PD	Yes	No	XX	XX
SOLAPAK.PRG	Com.	Yes	No	XX	XX
ARAM.TOS	Com.	Yes	No	XX	XX
INTRAMOK.ACC	PD	Yes	No	XX	XX
RAM380	PD	Yes	No	XX	XX
RAM512	PD	Yes	No	XX	XX
ETERNAL	PD	Yes	Yes	Yes	Yes
YARD	PD	Yes	Yes	Yes	Yes

Auto Loaders

As alluded to earlier, when working with a set of files (such as a compiler) over a period of time, manually loading files into the RAM disk each time the computer is turned on becomes rather tedious. As a result, a number of auto-loaders have appeared that load the files automatically when the computer is booted. I found a total of six and there are undoubtedly more. They are

all public domain and are as follows:

NAME	SOURCE	COMMENT
RAMDLDR	DN #73	File names, source and destination can be independently specified.
AUTODISK	GENie	Copies entire floppy to RAM disk
ULTCOPY	DN #63	Copies entire floppy to RAM disk
COPY	CIS	
FDCPIER	GENie	Copies folders to RAM disk
FLO2RDSK	GENie	Copies folders to RAM disk

The name of the game here is flexibility: you want to be able to specify what the files are, where they are and where they are to go. RAMDLDR satisfies all these criteria and works well with both ETERNAL and RAMDSKIM.

TURBO-BASIC THE FIRST REPORT

BY KEN WARD

THE NORWICH USERS GROUP

NORWICH, ENGLAND

ISSUE 24, FEBRUARY 1987

The biggest problem we've had up to now with alternative Basic's for the Atari has been the cost. OSS have produced an excellent range of language cartridges but they have been very expensive. Now TURBO-BASIC has changed all that. It's a low cost extended basic which not only offers a fantastic range of new

commands, but also speeds up all your existing ATARI Basic programs.

When we offered it as part of the "World of Atari" collection, we had only had time for a quick look at it. Here is our first report.

Testing It's Speed

TURBO-BASIC flies! It runs all ATARI BASIC programs 3 to 5 times faster! To try and assess the speed difference, we ran a simple test program...

```
10 FOR X=0 TO 2:POKE 18*X,0:NEXT X
20 FOR X=1 TO 1000
30 KEN=1256*22/7
40 NEXT X
50 ? PK(18)
```

This program was tried in both ATARI and TURBO basics, with different line 30's. In most cases TURBO basic was at least 3 times faster. We also tagged the routine to the front of a lengthy program, and a GOSUB to the end of program as line 30. TURBO-BASIC came out of this test an incredible 11 times faster!!

In fact, it's speed seems to be the main problem you'll have running ATARI BASIC Programs in TURBO BASIC! In some programs you will have to add delay loops to slow it down!

Problems With Bad Programming

I have come across one program that was a bit of trouble, but that was due to poor programming(which was surprising, because it was an Analog program!).

~~~~~  
**In the end, it prompted  
me to write my first  
routine in Turbo-Basic...**  
~~~~~

In the initializing section of the program there was the usual modifying of the display list by using:

```
DLIST=PEEK(560)+256*PEEK(561)
```

and then at the end of the init he had added another mod. By POKEing directly into where the display list would have been in ATARI BASIC instead of using the DLIST pointer!

The other problem was more intriguing. It centered around a loop like this...

```
10 POKE 764,255:POKE 53279,10
20 IF PEEK(764)=20 THEN 100
30 IF PEEK(764)=22 THEN 200
40 IF PEEK(53279)=5 THEN 300
50 IF PEEK(53279)=4 THEN 400
60 GOTO 10
```

It worked OK in ATARI BASIC, but in TURBO it popped straight out of the loop at line 40 even though the SELECT key had not been touched! We found that adding a short delay loop at line 15 allowed the loop to work correctly, as did POKEing 53279 with 8 in line 10, which is the correct value to clear the CONSUL keys.

If you come across any more examples - let us know.

TURBO Basic Memory Map

TURBO BASIC is a full 16k of code, yet it gives you another 1.5k of free memory over ATARI BASIC!

The bulk of TURBO BASIC is hidden under the Operating System ROM at the top of memory. The VBLANK routine has been modified to flip between the twinned memory blocks, allowing access to both areas.

The rest of TURBO BASIC sits in the block from 8320 (\$2080) to 13864 (\$9018). This is in the area normally used by DOS (and DUP when loaded), which explains why after calling DOS you cannot go back to TURBO BASIC. Which in turn explains why DOS commands have been added to the language.

Note that because of the re-arrangement of memory, the area occupied by the screen and display list at the top, and the variable tables, etc. at the bottom, are in new positions. Providing you use the pointers to find their new locations you'll be OK.

Formatting Disks

The only useful DOS command that is missing from TURBO BASIC is FORMAT. However, if you do get stuck and need to format another disk - the XIO commands still work.

XIO 254,#1,0,0,"D:" formats in the default drive format. If you have a 1050 and you need to format in single density use 253.

Changing Variable Names

The major problem I've found with my own programs is that I have been using variable names that are commands in TURBO BASIC! Names like MOVE, TEXT, DIR, MOD, DEC and HEX\$ are among my favorites! And of course it means I've had to rename them to stop TURBO BASIC erroring out.

Going through the programs modifying every occurrence of a name can be time consuming if there is a lot of them. One way out is to use a word processor in "Search and Replace" mode, but that means LISTing the program out and booting in the word processor. Again time consuming. In the end it prompted me to write my first routine in TURBO BASIC...

Type in the program and LIST it to disk. You then load the program you need to modify, ENTER the Renamer routine, and run it with G.32000

How It Works

In line 32070 we find the length of the variable table and dimension KEN\$ accordingly. The next two lines fill KEN\$ with complete list of variables.

The end of a variable is marked by being an inverse character, so in the next loop, which prints all the variables on the screen, we check for an inverse character at line 32120, and convert it before printing it. If there are a lot of variables, use CONTROL-I to freeze/unfreeze the screen.

You are then asked for the variable you want to rename.

Include the '(' if it's an array, and the '\$' if it's a string.

By printing the name you've typed in and positioning the cursor before getting an input, saves you the bother of typing in the complete name. Just modify one or two letters and hit RETURN.

A check is then made to make sure the names are the same length. (This subroutine only modifies the particular name - it doesn't re-write the complete variable table).

The last character of our first input is then inversed before using the INSTR command to find it's position in the variable table. If x=0 then you've tried to modify a

variable that doesn't exist!

If all is well, the last character of the new name is inversed before using the MOVE command to move the new name into the table area.

And that's all there is to it. Don't forget the programs you modify must be SAVED files. LISTed files won't put the names into the variable table in the first place - the lines will just error out.

We haven't bothered with TYPO codes on this one - the easiest way to check it is to run it by itself (after you've LISTed out a copy to disk, of course!). After you've run the program, LIST it on the screen to check the changed names.

```
32000 REM *****
32010 REM * VARIABLE RENAMER FOR *
32020 REM * USE WITH TURBO BASIC *
32030 REM * KEN WARD 8th Jan 87 *
32040 REM * NORWICH USERS GROUP *
32050 REM *****
32060 REM
32070 CLR :CLS :N=DPEEK(132)-DPEEK(130):DIM KEN$(N),VAR$(30),NXT$(30)
32080 KEN$=" ":KEN$(N)=" ":KEN$(2)=KEN$
32090 MOVE DPEEK(130),ADR(KEN$),N
32100 POSITION 2,0
32110 FOR X=1 TO LEN(KEN$)
32120 Y=ASC(KEN$(X,X)):IF Y>127 THEN Y=Y-128:? CHR$(Y):GOTO 32140
32130 ? CHR$(Y);
32140 NEXT X
32150 ? "What is the name of the variable you":? "want to rename ":INPUT VAR$
32160 ? "Please type in new name - REMEMBER -":? "name must be same length!":? "
":VAR$;:POSITION 2,PEEK(84)
32170 INPUT NXT$:IF LEN (NXT$)<>LEN(VAR$) THEN ? "<-" :GOTO 32160
32180 Y=ASC(VAR$(LEN(VAR$)))+128:VAR$(LEN(VAR$))=CHR$(Y)
32190 X=INSTR(KEN$,VAR$):IF X=0 THEN ? "<-" :GOTO 32150
32200 Y=ASC(NXT$(LEN(NXT$)))+128:NXT$(LEN(NXT$))=CHR$(Y)
32210 MOVE ADR(NXT$),DPEEK(130)+X-1,LEN(NXT$)
```

President's Column

As I look over the membership list, it comes to mind how different each and everyone of us really are. The diversity of professions represented by our members is quite unique. Our common thread, however, is our ownership of ATARI Computers.

~~~~~  
**Your participation is  
more than essential,  
it's critical!**  
~~~~~

This diversity of professions spills over into computer interests as well. It is difficult to program an effective agenda which will interest everyone all the time. We on the Executive Board want to make our club the best it can possibly be and WE are willing to take the time to do it, and our membership has what it takes to make it happen, provided you will step forward and share your abilities and interests with the rest of us.

Unfortunately, we are one step behind. We are missing the most vital element...YOU! We miss your attendance at our meetings, your newsletter articles, and your input on

items of interest on the BBS. We would like to see your latest programming effort, or your most recent hard or software purchase, show us how you mastered level 5 of Revenge of The Goop from Planet Xyphus.

It's our club, yours and mine. Your participation is more than essential, its critical! We can't be effective without it. Through the mutual sharing of information and ideas, we all benefit. Our club doesn't need joiners. We need interested, active members. Lets take this final step and make STATUS what it ought to be, a truly active club in every sense of the word.

Here's an idea for you to think about... Connect your VCR to your computer and record a "show" to be replayed at a club meeting. This would eliminate the impossibility of some demos at meetings (due to hard/software limitations) and would allow you to give a demo of first class quality.

Opinions expressed in this publication are those of the individual authors and do not necessarily express or reflect the opinions of the Southside Tidewater Atari Technical Users Society (STATUS). Some material contained herein may have been taken from Bulletin Board Systems and/or Newsletters of other groups and should not be construed as fact.

The material herein may be copied freely provided that full credit is given to the original author and the Southside Tidewater Atari Technical Users Society. STATUS is in no way affiliated with ATARI Corp.

Please address all Newsletter correspondence to:

STATUS
Newsletter Exchange
4836 Honeygrove Road
Virginia Beach, VA 23455
(804) 499-6021

Meetings: STATUS meetings are held on the first and third Thursdays of the month at the 7-UP Bottling Company, 5700 Ward Avenue, in Virginia Beach at 7:30 p.m. All interested parties are welcome to attend.

Newsletter Articles:

Submitted articles are preferred as disk text files, but will be gratefully accepted as hard copy (including handwriting) if you do not have a disk drive. If you have a modem, you can upload your articles to the Editor by calling the STATUS BBS at 468-1096. Articles may be submitted at any time, but will probably not make that month's Newsletter if submitted less than one week prior to the first meeting of the month.

STATUS

Newsletter Exchange
4836 Honeygrove Road
Virginia Beach, VA 23455
(804) 499-6021